Remarks

Claims 1-21 and 26-29 remain in the application. Claims 22-25 are hereby canceled without prejudice to facilitate allowance of the remaining claims. Claims 2, 9, 10, 12-21 and 26-27 are hereby amended. No new matter is being added.

Claim Objections

Original claim 2 is hereby amended to insert "and" in line 3 after "priority register;" so as to overcome the objection to claim 2.

Original claims 9, 10, 12 and 13 are hereby amended such that the acronyms IPF and PIC are no longer present in those claims. As such, applicants respectfully submit that the objection to these claims are overcome.

Claim Rejections -- 35 U.S.C. 112

Original claims 5 and 18 was rejected under 35 U.S.C. 112, first paragraph, in regards to the enablement of "storing the shadow copy in the low latency cache memory within the microprocessor if the task priority register is accessed frequently." (Item 3 of the office action.) Applicants respectfully traverse this objection.

Applicants respectfully point to the original application at page 8, line 33 through page 9, line 5 for support of the above-cited claim language. For convenience of reference, this portion of the specification is reproduced below.

> ... If reads of the interrupt mask occur with sufficient frequency, then the memory location of the shadow copy will be in local cache memory on the microprocessor, instead of in main memory. If, for example, the shadow copy is located in the first level cache, then only a single cycle would be needed to access the value therein, as opposed to the 36 cycle latency of the conventional method.

Applicants respectfully submit that the above-cited portion of the specification provides sufficient support for the claim language in original claims 5 and 18. Hence, applicants respectfully submit that this rejection is now overcome.

Claim Rejections -- 35 U.S.C. 103

Original claims 1-4, 6, 8-11, 14-17, 19, 21, and 26-29 were rejected under 35 U.S.C. 103 as being unpatentable over Williams et al in view of "Intel Itanium Processor Family Interrupt Architecture Guide." Applicants respectfully traverse this rejection.

Original claim 1 recites as follows.

1. A method of reducing access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor, the method comprising:

receiving a command to write an **interrupt mask** value to the task priority register;

writing the **interrupt mask** value to the task priority register; and

writing the **interrupt mask** value into a shadow copy of the task priority register,

wherein the shadow copy is written each time that the task priority register is written.

(Emphasis added.)

As seen from the above, claim 1 recites various steps relating to an "**interrupt mask** value". (Emphasis added.)

As is known in the microprocessor art, an **interrupt** (also called an **interrupt request**, or **interrupt vector**) is typically stored in an interrupt register. A general definition for such an interrupt is given, for example, by www.webopedia.com as follows.

Interrupt
(n.) A signal informing a program that an event has occurred. When a program receives an interrupt signal, it takes a specified action (which can be to ignore the signal). Interrupt signals can cause a program to suspend itself temporarily to service the interrupt.

Interrupt signals can come from a variety of sources. For example, every keystroke generates an interrupt signal. Interrupts can also be generated by other devices, such as a printer, to indicate that some event has occurred. These are called *hardware interrupts.* Interrupt signals initiated by programs are called *software interrupts*. A software interrupt is also called a *trap* or an *exception.*

PCs support 256 types of software interrupts and 15 hardware interrupts. Each type of software interrupt is associated with an *interrupt handler* -- a routine that takes control when the interrupt occurs. For example, when you press a key on your keyboard, this triggers a specific interrupt handler. The complete list of interrupts and associated interrupt handlers is stored in a table called the *interrupt vector table*, which resides in the first 1 K of addressable memory.

Also see the list of IRQ numbers in the Quick Reference section of Webopedia.

(Retrieved from http://www.webopedia.com/ on August 23, 2007.)

In contrast, an **interrupt mask** may be **applied to interrupt vectors** so as to determine whether or not that interrupt is to be processed or not. For example, the definition of "interrupt mask" retrieved from http://www.pcmag.com/ on August 23, 2007 states as follows. "An internal switch setting that controls whether an interrupt can be processed or not."

As a further example, the background section of U.S. Patent No. 6,175,890, explains the difference between interrupt requests and interrupt masks as follows.

In a system where a microprocessor is connected to a plurality of external devices, a plurality of **interrupt requests** for accessing the microprocessor may occur simultaneously. These requests may or may not be accepted depending on how the microprocessor is executing a current process.

> Acceptance of requests is controlled by an **interruption mask** flag (hereinafter, simply referred to as a mask flag) set in a mask register usually provided in the microprocessor. For example, an interruption is enabled when the mask flag is set to 0 and is disabled when the mask flag is set to 1. Hereinafter, a "mask level state" will refer to a status of a microprocessor process characterized by a unique set of mask flags.
>
> (Column 1, lines 13-25.)

Per the office action, original claim 1 was rejected based on the assumption that the "interrupt status value" of Williams et al. may be reasonably read onto the "interrupt mask value" which is recited multiple times in claim 1. Applicants respectfully traverse this assumption.

Williams et al. recites "copying an **interrupt status value** from its **interrupt register**." (Abstract of Williams et al., emphasis added.) As discussed above, the value stored in the interrupt register is generally the **interrupt request** (also known as the "interrupt vector" or simply the "interrupt"), **not the interrupt mask**.

Therefore, applicants respectfully submit that the "interrupt status value" of Williams et al does not read on the "interrupt mask value" of claim 1.

Applicants further respectfully submit that the "interrupt register" cited in Williams et al does not read on the "task priority register" of claim 1. In particular, while the interrupt register of Williams holds interrupt requests, the task priority register of claim 1 holds interrupt masks.

Therefore, applicants respectfully submit that the cited portions of Williams et al. does not teach any of the limitations of claim 1, as the cited portions of Williams et al. do not pertain to the claimed **interrupt mask** values stored in a **task priority register**.

In addition, **applicants respectfully submit that the "Intel Itanium Processor Family Interrupt Architecture Guide" does not teach the requirement of original claim 1 of "writing the interrupt mask value into a shadow copy of the task priority register, wherein the shadow copy is written each time that the task priority register is written."**

Therefore, for at least the above-discussed reasons, applicants respectfully submit that claim 1 overcomes this rejection.

Claims 2-4, 6, 8-11 depend from claim 1. Hence, applicants respectfully submit that claims 2-4, 6, 8-11 now overcome this rejection for at least the same reasons as discussed above in relation to claim 1. Further reasons for the allowability of these dependent claims are given below.

Claim 2 recites further limitations relating to **interrupt mask** values. In contrast, the cited portions of Williams et al in relation to claim 2 pertain to **interrupt status** values, which are not interrupt mask values. Therefore, applicants respectfully submit that this is a further reason by which claim 2 overcomes this rejection.

Claim 3 recites further limitations relating to the shadow copy of the **interrupt mask** value from the **task priority register**. In contrast, the cited portions of Williams et al in relation to claim 3 pertain to copying the **interrupt status** value from the **interrupt register**. Therefore, applicants respectfully submit that this is a further reason by which claim 3 overcomes this rejection.

Claim 4 recites further limitations relating to the shadow copy of the **interrupt mask** value from the **task priority register**. In contrast, the cited portions of Williams et al in relation to claim 3 pertain to copying the **interrupt status** value from the **interrupt register**. Therefore, applicants respectfully submit that this is a further reason by which claim 4 overcomes this rejection.

Claim 8 recites "a latency of reading from the **task priority register** is substantially reduced." In contrast, the cited portions of Williams et al do not pertain to such a register.

Amended claim 9 now recites "wherein **data in the task priority register reflects a level of priority of tasks being performed by the microprocessor**." (Emphasis added.) This claim language is supported on page 3, lines 2-3 of the original application. In contrast, the cited portions of Williams et al do not pertain to such a register.

Claims 14-17, 19 and 21 were rejected under "the same rationale" as claims 1-4, 6 and 8 discussed above. Applicants respectfully traverse these claim rejections for at least the same reasons discussed above in relation to claims 1-4, 6, and 8, respectively.

Like claim 1, claim 26 recites limitations pertaining to an "interrupt mask value" and a "task priority register". For reasons discussed above in relation to claim 1, **applicants respectfully submit that the "interrupt status value" of Williams et al does not read on the "interrupt mask value" of claim 26. Applicants further respectfully submit that the "interrupt register" cited in Williams et al does not read on the "task priority register" of claim 26.**

In addition, **applicants respectfully submit that the "Intel Itanium Processor Family Interrupt Architecture Guide" does not teach the requirement of claim 26 of "reading the interrupt mask value from the shadow copy at a memory location, instead of from the task priority register itself."**

Similarly, claim 27 recites limitations pertaining to an "interrupt mask value" and a "task priority register". For reasons discussed above in relation to

claim 1, **applicants respectfully submit that the "interrupt status value" of Williams et al does not read on the "interrupt mask value" of claim 27. Applicants further respectfully submit that the "interrupt register" cited in Williams et al does not read on the "task priority register" of claim 27.**

In addition, **applicants respectfully submit that the "Intel Itanium Processor Family Interrupt Architecture Guide" does not teach the requirement of claim 27 of "microprocessor-executable code configured to read the interrupt mask value from the shadow copy at a memory location, instead of from the task priority register itself."**

Original claim 28 also recites limitations pertaining to an "interrupt mask value" and a "task priority register". For reasons discussed above in relation to claim 1, **applicants respectfully submit that the "interrupt status value" of Williams et al does not read on the "interrupt mask value" of claim 28. Applicants further respectfully submit that the "interrupt register" cited in Williams et al does not read on the "task priority register" of claim 28.**

In addition, **applicants respectfully submit that the "Intel Itanium Processor Family Interrupt Architecture Guide" does not teach the requirements of claim 28 that "the memory system holds data including an operating system and shadow copies of the TPRs, and wherein the operating system includes executable-code for reading the interrupt mask values from the shadow copies and for maintaining the shadow copies."**

Original claim 29 also recites limitations pertaining to an "interrupt mask value" and a "task priority register". For reasons discussed above in relation to claim 1, **applicants respectfully submit that the "interrupt status value" of Williams et al does not read on the "interrupt mask value" of claim 29. Applicants further respectfully submit that the "interrupt register" cited in Williams et al does not read on the "task priority register" of claim 29.**

In addition, **applicants respectfully submit that the "Intel Itanium Processor Family Interrupt Architecture Guide" does not teach the**

requirements of claim 29 that "upon receiving a command to write an interrupt mask value to the task priority register, writing the interrupt mask value to the task priority register without performing a serialization directly thereafter" and "upon receiving an interrupt, performing the serialization and reading an interrupt vector register, wherein a spurious indicator is returned if the interrupt is maskable."

Claim Rejections -- 35 U.S.C. 103

Claims 5, 7, 18 and 20 were rejected under 35 U.S.C. 103 as being unpatentable over Williams et al in view of "Intel Itanium Processor Family Interrupt Architecture Guide" and further in view of Demharter. Applicants respectfully traverse this rejection.

Claims 5 and 18 recites limitations pertaining a "task priority register". For reasons discussed above in relation to claim 1, **applicants respectfully submit that the "interrupt register" cited in Williams et al does not read on the "task priority register" of claims 5 and 18.**

In addition, **applicants respectfully submit that neither the "Intel Itanium Processor Family Interrupt Architecture Guide" nor Demharter teach the requirement of claims 5 and 18 that "if the task priority register is accessed frequently, then the shadow copy is stored in low-latency cache memory within the microprocessor."**

Claims 7 and 20 also recite limitations pertaining a "task priority register". For reasons discussed above in relation to claim 1, **applicants respectfully submit that the "interrupt register" cited in Williams et al does not read on the "task priority register" of claims 7 and 20.**

In addition, **applicants respectfully submit that neither the "Intel Itanium Processor Family Interrupt Architecture Guide" nor Demharter**

teach the requirement of claims 7 and 20 that "a latency of writing to the task priority register is substantially reduced."

Conclusion

For the above-discussed reasons, applicant believes that the pending claims, as amended, now overcome all the objections and rejections of the latest office action. Favorable action is respectfully requested.

If for any reason an insufficient fee has been paid, the Commissioner is hereby authorized to charge the insufficiency to Deposit Account No. 08-2025 (Hewlett Packard).

Respectfully Submitted,

Dated: August 24, 2007

James K. Okamoto, Reg. No. 40,110
Okamoto & Benedicto LLP
P.O.Box 641330
San Jose, CA 95164-1330
Tel: (408) 436-2111
Fax: (408) 436-2114

CERTIFICATE OF MAILING			
I hereby certify that this correspondence, including the enclosures identified herein, is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below. If the Express Mail Mailing Number is filled in below, then this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service pursuant to 37 CFR 1.10.			
Signature:			
Typed or Printed Name:	James K. Okamoto	Dated:	August 24, 2007
Express Mail Mailing Number (*optional*):			